(19) Europäisches Patentamt
Europ an Pat nt Offic
Office européen des brevets

(11) EP 0 803 811 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
29.10.1997 Bulletin 1997/44

(51) Int Cl.[6]: **G06F 9/46**, G06F 9/445

(21) Application number: **97302769.1**

(22) Date of filing: **23.04.1997**

(84) Designated Contracting States:
DE FR GB

(30) Priority: **23.04.1996 US 636706**

(71) Applicant: **SUN MICROSYSTEMS, INC.**
Mountain View, CA 94043 (US)

(72) Inventors:
- **Wollrath, Ann, M.**
Groton, Massachusetts 01450 (US)
- **Waldo, James, H.**
Dracut, Massachusetts 01826 (US)
- **Riggs, Roger**
Burlington, Massachusetts 01803 (US)

(74) Representative: **Johnson, Terence Leslie**
Edward Evans & Co.
Chancery House
53-64 Chancery Lane
London WC2A 1SD (GB)

(54) System and method for stub retrieval and loading

(57) A stub retrieval and loading subsystem is disclosed for use in connection with a remote method invocation system. The stub retrieval and loading subsystem controls the retrieval and loading ofa stub for a remote method, into an execution environment, to facilitate invocation of the remote method by a program executing in the execution environment. The stub retrieval subsystem includes a stub retriever for initiating a retrieval of the stub and stub loader for, when the stub is received by the stub retriever, loading the stub into the execution environment, thereby to make the stub available for use in remote invocation of the remote method. In one embodiment, the stub retrieval and loading subsystem effects the retrieval and loading for a program operating in one address space provided by one computer, of stub class instances to effect the remote invocation of methods which are provided by objects operating in another address space, which may be provided by the same computer or a different computer.

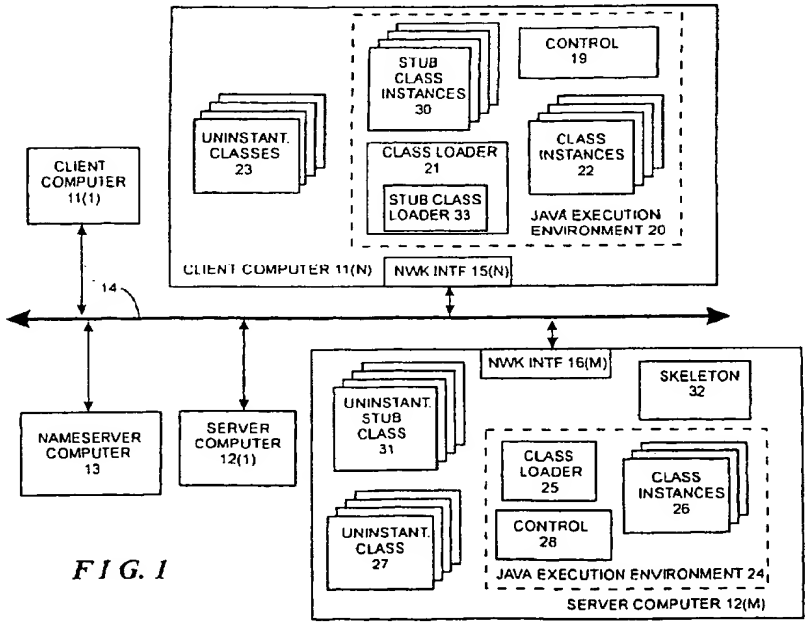


FIG. 1

EP 0 803 811 A2

taining, dynamic loading and use of "stub" information to enable a program operating in one address space to invoke processing of a remote method or procedure in another address space;

FIGs. 2 and 3 are flow charts depicting the operations performed by the arrangement depicted in FIG. 1, which is useful in understanding the invention, with FIG. 2 depicting operations performed in connection with obtaining and dynamic loading of the stub information and FIG. 3 depicting operations performed in connection with use of the stub information to invoke processing of the remote method or procedure.

DETAILED DESCRIPTION OF AN ILLUSTRATIVE EMBODIMENT

FIG. is a schematic diagram of a computer network 10 including an arrangement for facilitating dynamic loading of "stub" information to enable a program operating in one address space to remotely invoke processing of a method or procedure in another address space. With reference to FIG. 1, computer network 10 includes a plurality of client computers 11(1) through 11(N) (generally identified by reference numeral 11(n)), a plurality of server computers 12(1) through 12(M) (generally identified by reference numeral 12(m)), all of which are interconnected by a network represented by a communication link 14. In addition, the network 10 may include at least one nameserver computer 13, which may also be connected to communication link 14, whose purpose will be described below. As is conventional, at least some of the client computers 11(n) are in the form of personal computers or computer workstations, each of which typically includes a system unit, a video display unit and operator input devices such as a keyboard and mouse (all of which are not separately shown). The server computers 12(m) and nameserver computer 13 also typically include a system unit (also not separately shown), and may also include a video display unit and operator input devices.

The client computers 11(n), server computers 12 (m) and nameserver computer 13 are all of the conventional stored-program computer architecture. A system unit generally includes processing, memory, mass storage devices such as disk and/or tape storage elements and other elements (not separately shown), including network interface devices 15(n), 16(m) for interfacing the respective computer to the communication link 14. The video display unit permits the computer to display processed data and processing status to the operator, and an operator input device enables the operator to input data and control processing by the computer. The computers 11(n) and 12(m) and 13 transfer information, in the form of messages, through their respective network interface devices 15(n), 16(m) among each other over the communication link 14.

In one embodiment, the network 10 is organized in a "client-server" configuration, in which one or more computers, shown in FIG. 1 as computers 12(m), operate as servers, and the other computers, shown in FIG. 1 as computers 11(n) operate as clients. In one aspect, one or more of the server computers 12(m) may, as "file servers," include large-capacity mass storage devices which can store copies of programs and data which are available for retrieval by the client computers over the communication link 13 for use in their processing operations. From time to time, a client computer 11(n) may also store data on the server computer 12, which may be later retrieved by it (the client computer that stored the data) or other client computers for use in their processing operations. In addition, one or more of the server computers 12(m) may, as "compute servers," perform certain processing operations in response to a remote request therefor from a client computer 11(n), and return the results of the processing to the requesting client computer 11(n) for use by them (that is, the requesting client computers 11(n)) in their subsequent processing. In either case, the server computers may be generally similar to the client computers 11(n), including a system unit, video display unit and operator input devices and may be usable by an operator for data processing operations in a manner similar to a client computer. Alternatively, at least some of the server computers may include only processing, memory, mass storage and network interface elements for receiving and processing retrieval, storage or remote processing requests from the client computers, and generating responses thereto. It will be appreciated a client computer 11(n) may also perform operations described herein as being performed by a server computer 12(m), and similarly a server computer 12(m) may also perform operations described herein as being performed by a client computer 11(n).

The network represented by communication link 14 may comprise any of a number of types of networks over which client computers 11(n), server computers 12(m) and nameserver computers 13 may communicate, including, for example, local area networks (LANs) and wide area networks (WANs) which are typically maintained within individual enterprises, the public telephony system, the Internet, and other networks, which may transfer digital data among the various computers. The network may be implemented using any of a number of communication media, including, for example, wires, optical fibers, radio links, and/or other media for carrying signals representing information among the various computers depicted in FIG. 1. As noted above, each of the computers typically includes a network interface which connects the respective computer to the communications link 14 and allows it to transmit and receive information thereover.

The invention provides a system for facilitating the obtaining and dynamic loading of "stub" information to enable a program operating in one address space to invoke processing of a remote method or procedure in an-

cludes declarations for the complete set of interfaces for the particular remote uninstantiated class 27 which implements the remote method to be invoked, and also provides or invokes methods which facilitate accessing of the remote method(s) which are implemented by the remote class. The uninstantiated stub class 31, when it is instantiated and provided to the execution environment 20 of the client computer 11(n) as a stub class instance 30, effectively provides the information which is needed by the control module 19 of the execution environment 20 of the invoking Java program, so that, when a remote method that is implemented by its associated class is invoked by a Java program running in a particular execution environment, the remote method will be processed and the return value(s) provided to the invoking Java program. In one embodiment, the arrangement by which the stub class instance may be provided to the execution environment 20 is similar to that described in the Waldo, et al., patent application entitled "System and Method for Generating Identifiers for Uniquely Identifying Object Types for Objects Used in Processing of Object-Oriented Programs and the like", having the same priority date of the present application. A copy of the Waldo, et al., patent application is filed herewith.

In addition, the server computer 12(m) provides a skeleton 32, which identifies the particular classes and methods which have been exported by the server computer 12(m) and information as to how it (that is, the server computer 12(m)) may load the respective classes and initiate processing of the particular methods provided thereby.

When a class instance invokes a remote method maintained by a server computer 12(m), it will provide values for various parameters to the stub class instance 30 for the remote method, which values the remote method will use in its processing. If the remote method is implemented on the same computer as the invoking Java program, when the invoking Java program invokes a remote method, the computer may establish an execution environment, similar to the execution environment 20, enable the execution environment's class loader to load and instantiate the class which implements the method as a class instance similar to class instances 22, and process the remote method using values of parameters which are provided by the invoking class instance in the remote invocation. After processing of the method has been completed, the execution environment in which the remote method has been processed will provide the results to the stub class instance 30 for the remote method that was invoked, which, in turn, will provide to the particular class instance 22 which invoked the remote method.

Similar operations will be performed if client computer 11(n) and server computer 12(m) are implemented on different physical computers. In that case, in response to a remote invocation, the client computer 11 (n) that is processing the invoking class instance 22, under control of the control module 19 for the execution environment 10 for the invoking class instance 22, will use the appropriate stub class instance 30 to communicate over the network represented by the communication link 14 with the server computer 12(m) which implements the remote method to enable it (that is, the server computer 12(m)) to establish an execution environment 24 for the class which implements the remote method, and to use the class loader 25 to load an instance of the class as a class instance 26. In addition, the client computer 11(n), also using the appropriate stub class instance 30, will provide any required parameter values to the server computer 12(m) over the network 14. Thereafter, the server computer 12(m) will process the remote method using parameter values so provided, to generate result value(s) which are transferred over the network to the client computer 11(n), in particular to the appropriate stub class instance 30. The client computer 11(n) will, after it receives the result value(s) from the network, provide them to the invoking class instance 22 for its processing.

In any case, when the control module 19 of the client computer's execution environment 20 determines that a reference to the remote object has been received, if it determines that the stub class instance 30 is not present when it receives the reference, it will attempt to obtain the stub class instance 30 from, for example, the server computer 12(m) which implements the remote method, and enable the stub class instance 30 to be dynamically loaded in the execution environment 20 for the invoking class instance 22. A reference to the remote object may be received, for example, either as a retum value of another remote method invocation or as a parameter that is received during another remote method invocation. The stub class instance may be dynamically loaded into the execution environment in a manner similar to that used to load class instances 22 in the execution environment 22. The execution environment 20 is provided with a stub class loader 33 which, under control of the control module 19, will attempt to find and load the stub class instances 30 as required by the class instances 22 processed in the execution environment. The location of a particular server computer 12(m) that maintains the class that implements a method to be invoked remotely may be included in the call from the invoking class instance or may be made known to the stub class loader 33 through another mechanism (not shown) maintained by the client computer 11(n).

However, if the stub class loader 33 is not otherwise notified of which server computer 12(m) maintains the class which implements a method which may be invoked remotely, it may use the nameserver computer 13 to provide that identification. The identification may comprise any identifier which may be used to identify a server computer 12(m) or other resource which is available on the network 14 and to which the server computer 12(m) can respond. Illustrative identifiers include, for example, a network address which identifies the server computer and/or resource, or, if the network 14 is or in-

stance 30 into execution environment 20 for the class instance 21 which initiated the remote method invocation call in step 100 (step 104). After the stub class instance 30 for the referenced remote method has been loaded in the execution environment, the method can be invoked as will be described below in connection with FIG. 3.

Returning to step 102, if the control module 19 determines that the invocation from the class instance 22 did not include a resource locator to identify the server computer 12(m) or other resource which maintains the class for the method to be invoke, and further that it (that is, the control module 19) or the stub class loader 33 is not otherwise provided with such a resource locator, a "class not found" exception may be indicated, at which point the control module 19 may call an exception handler. The exception handler may perform any of a number of recovery operations, including, for example, merely notifying the control module 19 that the remote method could not be located and allow it to determine subsequent operations.

Alternatively, the control module 19 may attempt to obtain a resource locator from the nameserver computer 13 or other resource provided by the network 14 (generally represented in FIG. 1 by the nameserver computer 13), using a call to, for example, a default stub class instance 30. The call to the default stub class instance 30 will include an identification of the class and method to be invoked and the name ofthe nameserver computer 13(m). Using the default stub class instance 30, the control module 19 will enable the computer 11 (n) to initiate communications with nameserver computer 13 to obtain an identifier for a server computer 12(m) which maintains the class and method to be invoked (step 110). The communications from the default stub class instance 30 will essentially correspond to a remote method invocation, with the method enabling the nameserver computer to provide the identification for the server computer 12(m), if one exists associated with the class and method to be remotely invoked, or alternatively to provide an indication that no server computer 12 (m) is identified as being associated with the class and method. During the communications in step 110, the default stub class interface 30 will provide, as a parameter value, the identification of class and method to be invoked.

In response to the communications from the default stub class instance 30, the nameserver computer 13 will process the request as a remote method (step 111), with the result information comprising the identification for the server computer 12(m), if one exists that is associated with the class and method to be remotely invoked, or alternatively an indication that no server computer 12 (m) is identified as being associated with the class and method. After finishing the method, the nameserver computer 13 will initiate communications with the default stub class instance 30 to provide the result information to the default stub class instance 30 (step 112).

After receipt of the result information from the nameserver computer 13, the default stub class instance, under control of the control module 19, will pass result information to the stub class loader 33 (step 113). Thereafter, the stub class loader 33 determines whether the result information from the nameserver computer comprises the identification for the server computer 12 (m) or an indication that no server computer 12(m) is identified as being associated with the class (step 114). If the stub class loader 33 determines that the result information comprises the identification for the server computer 12(m), it (that is, the stub class loader 33) will return to step 101 to initiate communication with the identified server computer 12(m) to obtain stub class instance for the class and method that may be invoked. On the other hand, if the stub class loader 33 determines in step 114 that the nameserver computer 13 had provided an indication that no server computer 12(m) is identified as being associated with the class and method that may be invoked, the "class not found" exception may be indicated (step 115) and an exception handler called as described above.

As noted above, the stub class instance 30 retrieved and loaded as described above in connection with FIG. 2 may be used in remote invocation of the method. Operations performed by the client computer 11(n) in connection with remote invocation of the method will be described in connection with the flow chart in FIG. 3. As depicted in FIG. 3, when a class instance 22 invokes a method, the control module 19 may initially verify that a stub class instance 30 is present in the execution environment for remote method to be invoked (step 120). If a positive determination is made in step 120, the stub class instance 30 will be used for the remote invocation, and in the remote invocation will provide parameter values which are to be used in processing the remote method (step 121). Thereafter, the stub class instance 30 for the remote method that may be invoked will be used to initiate communications with the server computer 12(m) which maintains the class for the remote method (step 122), in the process, the passing parameter values which are to be used in processing the remote method will be passed. It will be appreciated that, if the server computer 12(m) which is to process the method is the same physical computer as the client computer 12(n) which is invoking the method, the communications can be among execution environments which are being processed within the physical computer. On the other hand, if the server computer 12(m) which is to process the method is a different physical computer from that of the client computer 12(n) which is invoking the method, the communications will be through the client computer's and server computer's respective network interfaces 15(n) and 16(m) and over the network 14.

In response to the communications from the stub class instance in step 122, the server computer 12(m), if necessary establishes an execution environment 24

ter to perform the processes described.

The processes may also be performed by electronic means.

Claims

1. For use in connection with a remote method invocation system, a stub retrieval and loading subsystem for controlling the retrieval and loading of a stub for a remote method into an execution environment to facilitate invocation of the remote method by a program executing in said execution environment, the stub retrieval subsystem comprising:

   A. a stub retriever for initiating a retrieval of said stub; and

   B. a stub loader for, when said stub is received by said stub retriever, loading said stub into said execution environment, thereby to make the stub available for use in remote invocation of said remote method.

2. A stub retrieval and loading subsystem as defined in claim 1 further including a remote method reference detector for detecting whether a remote method reference has been received in said execution environment, the stub retriever initiating retrieval of said stub when the remote method reference detector detects that a remote method reference has been received in said execution environment.

3. A stub retrieval and loading subsystem as defined in claim 1 further including a remote method invocation control for controlling invocation of said remote method, said stub retriever initiating retrieval of said stub when the remote method is invoked.

4. For use in connection with a remote method invocation method, a stub retrieval and loading method for facilitating the retrieval and loading of a stub for a remote method into an execution environment to facilitate invocation of the remote method by a program executing in said execution environment, the stub retrieval method comprising the steps of:

   A. a stub retrieval step for initiating a retrieval of said stub; and

   B. a stub loading step for, when said stub is received, loading said stub into said execution environment, thereby to make the stub available for use in remote invocation of said remote method.

5. For use in connection with a remote method invocation method, a signal for controlling the retrieval and loading of a stub for a remote method into an execution environment to facilitate invocation of the remote method by a program executing in said execution environment, the signal causing the computer to perform:

   A. a stub retrieval step for initiating a retrieval of said stub; and

   B. a stub loading step for, when said stub is received, loading said stub into said execution environment, thereby to make the stub available for use in remote invocation of said remote method.

6. A method of storing data on a recording medium, the method comprising storing data representative of a signal, which signal is for use in connection with a remote method invocation method, the signal being for controlling the retrieval and loading of a stub for a remote method into an execution environment to facilitate invocation of the remote method by a program executing in said execution environment, the signal causing the computer to perform:

   A. a stub retrieval step for initiating a retrieval of said stub; and

   B. a stub loading step for, when said stub is received, loading said stub into said execution environment, thereby to make the stub available for use in remote invocation of said remote method.

7. A method as defined in claim 4 or 6, or a signal as defined in claim 5 further including a remote method reference detection step for detecting whether a remote method reference has been received in said execution environment, the stub retrieval step including the step of initiating retrieval of said stub when a remote method reference has been received in said execution environment.

8. A method as defined in claim 4 or 6, or a signal as defined in claim 5 further including a remote method invocation control step for controlling invocation of said remote method, said stub retrieval step including the step of initiating retrieval of said stub when the remote method is invoked.

9. For use in connection with a remote method invocation system, a stub retrieval and loading computer program product for controlling a computer to, in turn, control the retrieval and loading of a stub for a remote method into an execution environment to facilitate invocation of the remote method by a program executing in said execution environment, the stub retrieval computer program product compris-

23. A stub retrieval and loading computer program product as defined in claim 22 further including remote method reference detector code devices for enabling said computer to detect whether a remote method reference has been received in said execution environment, the remote method reference including a remote method server identifier, the remote server identifier code devices enabling said computer to use the remote method server identifier as the server identification.

24. A stub retrieval and loading computer program product as defined in claim 22 further including remote method invocation control code devices for enabling said computer to provide a remote method invocation identification for controlling invocation of said remote method, the remote method invocation providing a remote method server identifier, the remote server identifier code devices enabling said computer to use the remote method server identifier as the server identification.

25. A stub retrieval and loading computer program product as defined in claim 22 the remote method invocation system further including a nameserver for providing a said server identification, said remote server identifier code devices enabling said computer to initiate communication with said nameserver to obtain the server identification for said remote method.

26. For use in connection with a remote method invocation system, a stub retrieval and loading subsystem for controlling the retrieval and loading of stub for a remote method into an execution environment to facilitate invocation of the remote method by a program executing in said execution environment, the stub retrieval subsystem comprising:

A. a computer; and

B. a control arrangement for controlling said computer, said control arrangement comprising:

i. a stub retrieval module for controlling said computer to initiate a retrieval of said stub; and

ii. a stub loader module for controlling said computer to, when said stub is received in response to said stub retrieval module, load said stub into said execution environment, thereby to make the stub available for use in remote invocation of said remote method.

27. A control arrangement for use in connection with a computer to control the retrieval and loading of a stub for a remote method into an execution environment to facilitate invocation of the remote method by a program executing in said execution environment, said control arrangement comprising:

i. a stub retrieval module for controlling said computer to initiate a retrieval of said stub; and

ii. a stub loader module for controlling said computer to, when said stub is received in response to said stub retrieval module, load said stub into said execution environment, thereby to make the stub available for use in remote invocation of said remote method.

28. A system for distributing code stored on a computer readable medium and executable by a computer, the code including a plurality of modules each configured to control the computer to facilitate the retrieval and loading of a stub for a remote method into an execution environment to facilitate invocation of the remote method by a program executing in said execution environment, said system comprising:

i. a stub retrieval module for controlling said computer to initiate a retrieval of said stub; and

ii. a stub loader module for controlling said computer to, when said stub is received in response to said stub retrieval module, load said stub into said execution environment, thereby to make the stub available for use in remote invocation of said remote method.

29. A signal according to any one of claims 5, 12 to 14, or 18 to 21, wherein the signal is recorded on a recording medium.

30. A signal according to claim 29, wherein the recording medium comprises a magnetic disc, a magnetic tape, an optical disc or an electronic memory device.

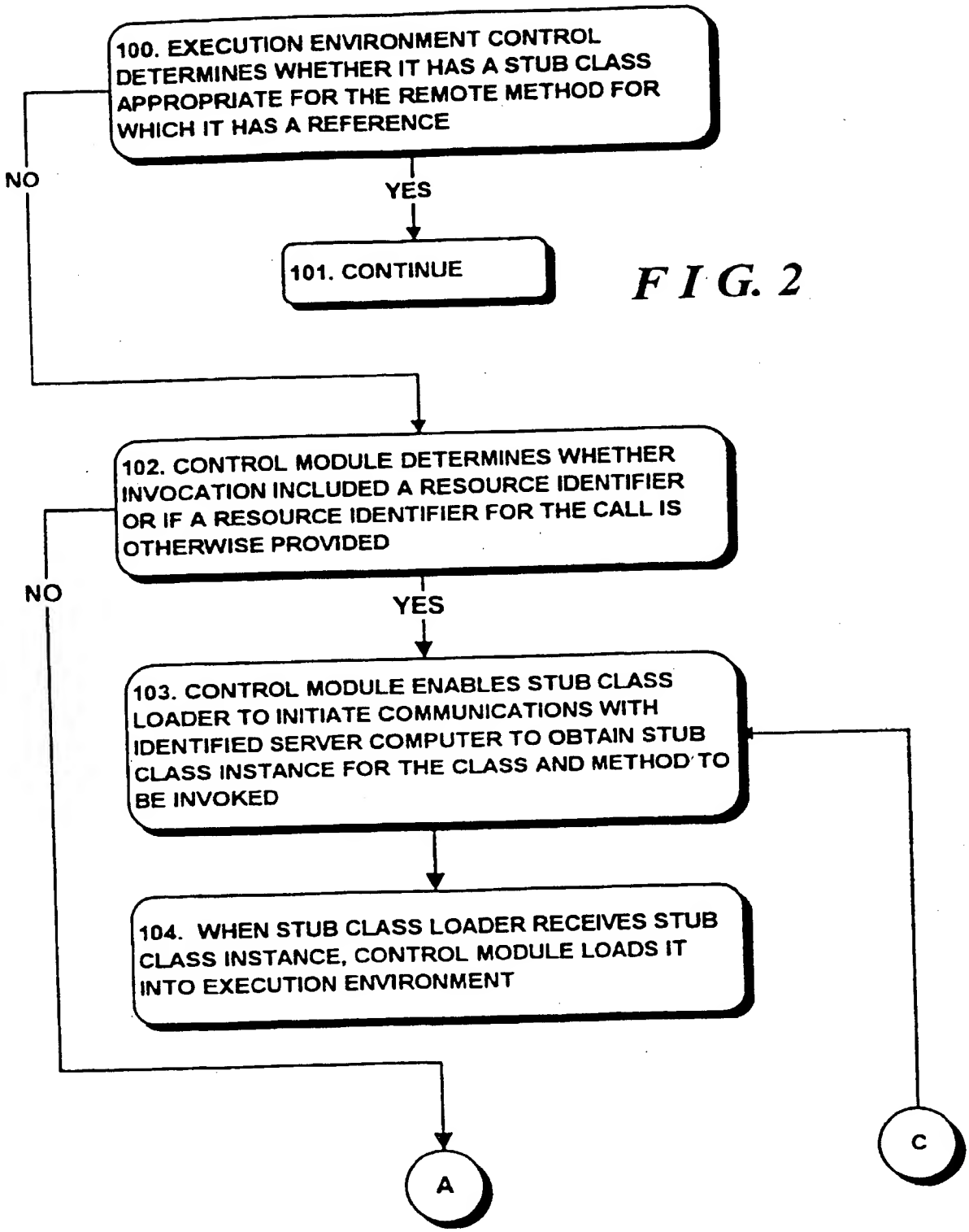
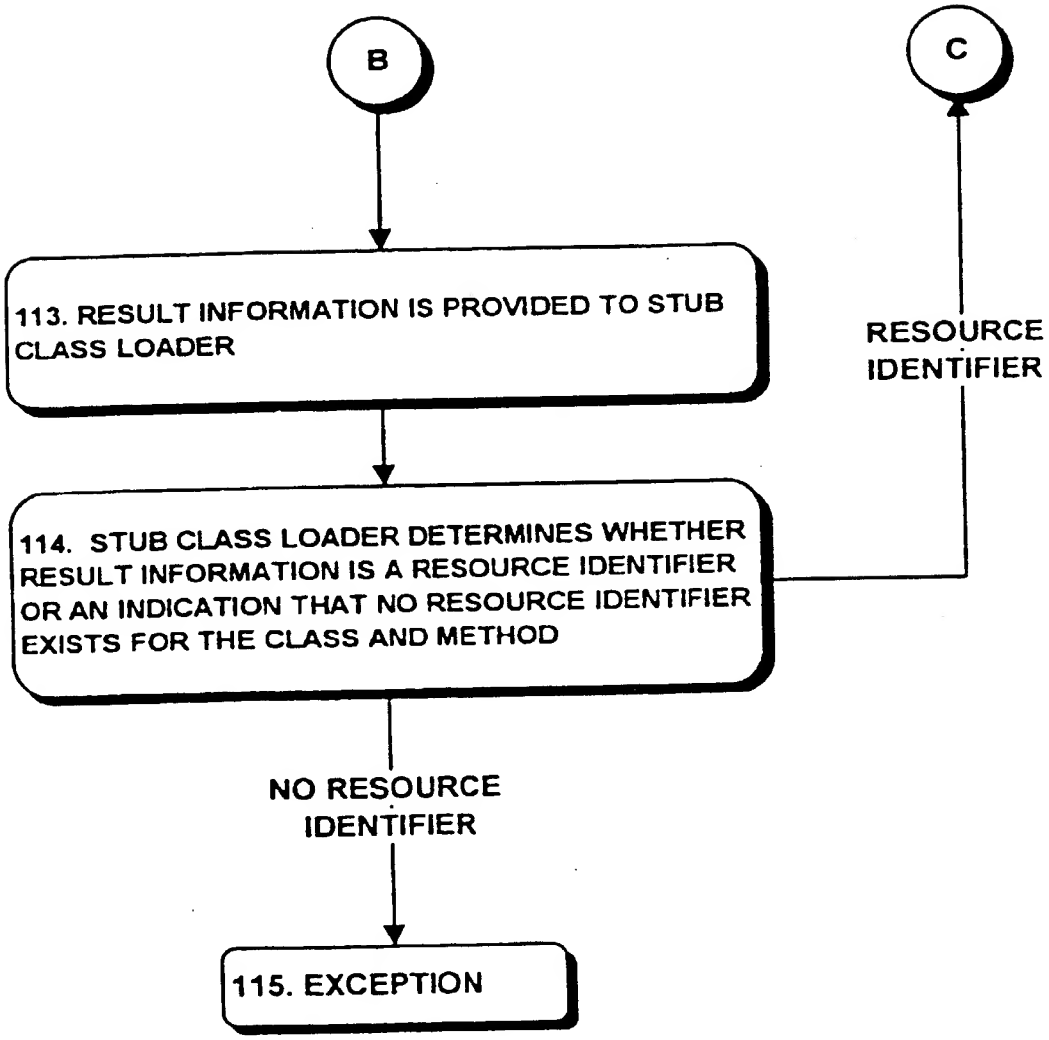
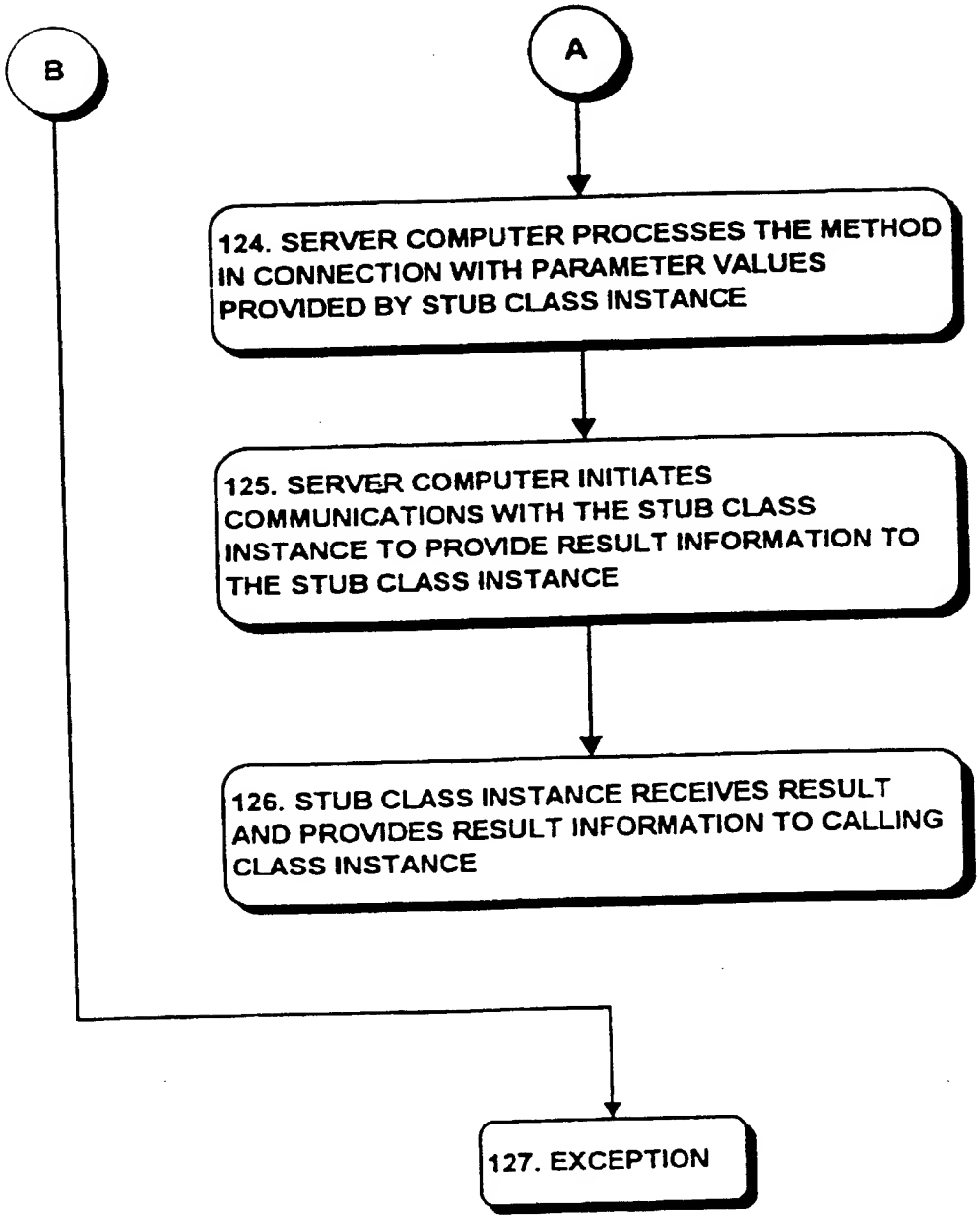


FIG. 2

F I G. 2 (CONT. B)



F I G. 3 (CONT. A)



(19) Europäisches Patentamt
Europ an Patent Office
Office europ´en des brev ts

(11) EP 0 803 811 A3

(12) EUROPEAN PATENT APPLICATION

(88) Date of publication A3:
06.12.2000 Bulletin 2000/49

(43) Date of publication A2:
29.10.1997 Bulletin 1997/44

(21) Application number: **97302769.1**

(22) Date of filing: **23.04.1997**

(51) Int Cl.[7]: **G06F 9/46**, G06F 9/445

(84) Designated Contracting States:
DE FR GB

(30) Priority: **23.04.1996 US 636706**

(71) Applicant: **SUN MICROSYSTEMS, INC.**
Mountain View, CA 94043 (US)

(72) Inventors:
- **Wollrath, Ann, M.**
  Groton, Massachusetts 01450 (US)
- **Waldo, James, H.**
  Dracut, Massachusetts 01826 (US)
- **Riggs, Roger**
  Burlington, Massachusetts 01803 (US)

(74) Representative: **Johnson, Terence Leslie**
Edward Evans & Co.,
Clifford's Inn,
Fetter Lane
London EC4A 1BX (GB)

(54) System and method for stub retrieval and loading

(57) A stub retrieval and loading subsystem is disclosed for use in connection with a remote method invocation system. The stub retrieval and loading subsystem controls the retrieval and loading ofa stub for a remote method, into an execution environment, to facilitate invocation of the remote method by a program executing in the execution environment. The stub retrieval subsystem includes a stub retriever for initiating a retrieval of the stub and stub loader for, when the stub is received by the stub retriever, loading the stub into the execution environment, thereby to make the stub available for use in remote invocation of the remote method. In one embodiment, the stub retrieval and loading subsystem effects the retrieval and loading for a program operating in one address space provided by one computer, of stub class instances to effect the remote invocation of methods which are provided by objects operating in another address space, which may be provided by the same computer or a different computer.

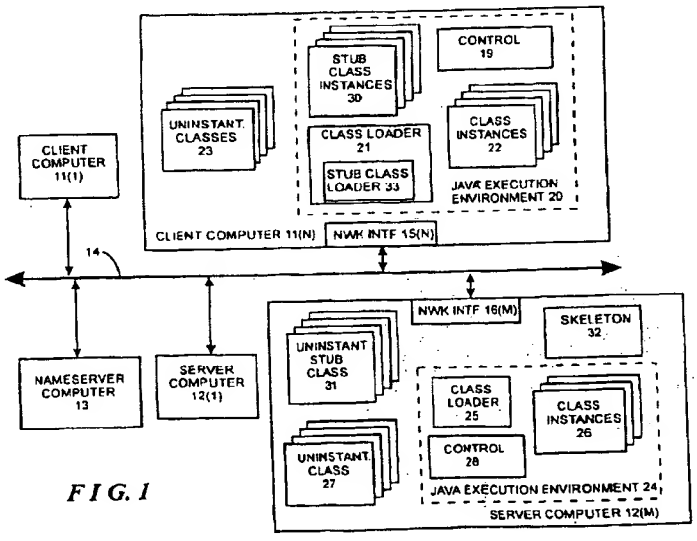


FIG. 1

EP 0 803 811 A3

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 97 30 2769

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

09-10-2000

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0497022 A	05-08-1992	DE 69131094 D	12-05-1999
		DE 69131094 T	29-07-1999
US 5452459 A	19-09-1995	NONE	
WO 9325962 A	23-12-1993	CA 2098418 A,C	19-12-1993
		DE 69220093 D	03-07-1997
		DE 69220093 T	04-12-1997
		EP 0646260 A	05-04-1995
		IL 105568 A	19-01-1996
		JP 8502841 T	26-03-1996
		US 5606493 A	25-02-1997

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82